When analyzing the running time of an algorithm, it's often useful to consider a more general form of the relationship between the input size N and the running time $T(N)$. If the running time of an algorithm is described by $T(N) \approx aN^b$, where a and b are constants, this form can help us determine the nature of the time complexity. Here's how you can analyze it:

1. **Measure the running time for different input sizes N:**
   - For $N = N_1$, measure $T(N_1)$.
   - For $N = N_2 = 2N_1$, measure $T(N_2)$.
2. **Compare the running times:**
   - Calculate the ratio $\frac{T(N_2)}{T(N_1)}$.

Steps in Detail

1. **Measure the Running Time:**
   - Let N_1 be the initial input size and $T(N_1)$ be the running time for this input size.
   - Let $N_2 = 2N_1$ be the doubled input size and $T(N_2)$ be the running time for this input size.
2. **Formulate the Relationship:**

   Given $T(N) \approx aN^b$:

   $$T(N_1) \approx aN_1^b$$

   $$T(N_2) \approx a(2N_1)^b = a \cdot 2^b \cdot N_1^b = 2^b \cdot aN_1^b$$
3. **Calculate the Ratio:** $\frac{T(N_2)}{T(N_1)} \approx \frac{2^b \cdot aN_1^b}{aN_1^b} = 2^b$

Determine the Exponent b

To find b, you can use the ratio calculated from your measurements: $b \approx \log_2\left(\frac{T(N_2)}{T(N_1)}\right)$

Example

Suppose you measure the running times and find the following:

- $T(N_1) = 2$ seconds for $N_1 = 1000$
- $T(N_2) = 16$ seconds for $N_2 = 2000$

Calculate the ratio: $\frac{T(2000)}{T(1000)} = \frac{16}{2} = 8$

Now, solve for b: $b \approx \log_2(8) = 3$

So, in this case, the running time $T(N)$ is approximately proportional to N^3, indicating a time complexity of $O(N^3)$.

General Procedure

1. **Measure** the running times for input sizes N_1 and $N_2 = 2N_1$.
2. **Calculate** the ratio $\frac{T(N_2)}{T(N_1)}$.
3. **Determine** b using $b \approx \log_2\left(\frac{T(N_2)}{T(N_1)}\right)$.

This approach helps you determine the polynomial order of the algorithm's time complexity when the running time can be approximated by $T(N) \approx aN^b$.